4-9-07

ATTORNEY DOCKET NO.
063170.6294

PATENT APPLICATION
09/905,533

O I P E
APR 06 2007
PATENT & TRADEMARK OFFICE

In the United States Patent and Trademark Office on Appeal from the Examiner to the Board of Patent Appeals and Interferences

In re Application of:	Myles Jordan
U.S. Patent Serial No.:	09/905,533
Filing Date:	July 14, 2001
Confirmation No.:	3486
Examiner:	Kevin R. Schubert
Group Art Unit:	2137
Title:	DETECTION OF DECRYPTION TO IDENTIFY ENCRYPTED VIRUS

CERTIFICATE OF MAILING
BY EXPRESS MAIL
Exp. Mail Receipt No. EV 732542656 US

I hereby certify that this communication is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" under 37 C.F.R. § 1.10 on the date indicated below and is addressed to Mail Stop: Appeal Brief - Patents, Commissioner For Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Willie Jiles
Willie Jiles

Date: April 6, 2007

MAIL STOP: APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

Corrected Appeal Brief

Appellant has appealed to the Board of Patent Appeals and Interferences ("Board") from the decision of the Examiner mailed March 3, 2006, finally rejecting all pending Claims 1-18. Appellant filed a Notice of Appeal on June 30, 2006, with the statutory fee of $500.00. Appellant submitted an Appeal Brief for consideration of the Board on September 14, 2006 with the statutory fee of $500.00. Appellant files this Corrected Appeal Brief in response to the Notice of Non-Compliant Appeal Brief dated March 30, 2007.

Conclusion

Appellant has demonstrated that the present invention, as claimed, is clearly distinguishable over the prior art cited by the Examiner. Therefore, Appellant respectfully requests the Board to reverse the final rejections and instruct the Examiner to issue a Notice of Allowance with respect to all pending claims.

Although Appellant believes that no fees are due, the Commissioner is hereby authorized to charge any fees or credit any overpayment to Deposit Account No. 02-0384 of Baker Botts, L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellant

[illegible]

Keiko Ichiye
Reg. No. 45,460
(214) 953-6494

Date: April 6, 2007

Correspondence Address:

Customer Number: **05073**

Real Party In Interest

This Application is currently owned by Computer Associates Think, Inc., as indicated by:

an Assignment recorded on February 11, 2002 from the inventor, Myles Jordan, to Computer Associates Think, Inc., in the Assignment Records of the PTO at Reel 012571, Frame 0484 (3 pages).

Related Appeals and Interferences

To the knowledge of Appellant's counsel, there are no known interferences or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.

Status of Claims

Claims 1-18 are pending in this Application, stand rejected pursuant to a Final Office Action mailed March 3, 2006 ("Final Office Action"), and are all presented for appeal. All pending claims are shown in the Claims Appendix, attached hereto, along with an indication of the status of those claims.

Status of Amendments

All amendments submitted by Appellant have been entered by the Examiner. The Examiner indicated in the Advisory Action mailed May 22, 2006 ("Advisory Action"), that the amendments presented in the Response to the Final Office Action have been entered. (Advisory Action, page 2.)

Summary of Claimed Subject Matter

The present invention relates to a method for detecting decryption of encrypted viral code in a subject file. (Page 3, lines 14-15.) Specifically, a list of memory regions that have been read and then modified during the emulation may be maintained. (Page 3, lines 22-23.) One of the listed memory regions may be determined to be larger than a predetermined size. (Page 9, lines 3-4.) A viral detection alarm is triggered in response to determining that one of the listed memory regions is larger than the predetermined size, where the viral detection alarm indicates detection of viral code. (Page 9, lines 7-9.)

FIG. 4 illustrates a computer system or computer 50 on which the potentially infected program or file may be executed. Computer system 50 includes a processor 51, memory 52, hard disk 53, removable storage drive 54 (for reading/accessing removable storage media, such as floppy disks, CDs, DVDs, etc.), a display 56, I/O devices 57 (for example, keyboard, mouse, microphone, speaker, etc.), and a wired or wireless connection to a network 58. The network can be, for example, a LAN, a WAN, an intranet, an extranet, the Internet, and/or any combinations of such networks. Computer 50 may be any of the computing devices/systems known in the art, such as, for example, a personal computer, a laptop, a workstation computer, a mainframe computer, a personal digital assistant (PDA), etc. (also referred to herein either separately or collectively as "computing device", "computer", "computer system", or "computing system"). (Page 6, lines 5-15.)

Executable code in a program or file that may be infected with a virus may reside on/in, for example, hard disk 53 and/or a removable storage medium that may be read/accessed through removable storage drive 54. Also, the executable code may be downloaded to the system through network 58. When executable code is analyzed according to the methodologies described herein, the file or program that the executable code or portion of executable code being analyzed is from is also referred to herein as the "subject file", as noted above. (Page 6, lines 16-22.)

Typically, decryption of viral code in a subject file is triggered when the program is executed or the file is opened. When the subject file is downloaded through a network, the download process may include opening potentially infected file so that detecting viral code in this scenario is also preferred. The methodologies of the present application are capable of detecting viral code in either scenario described above, as well as other scenarios. (Page 6, lines 23-28.)

Since legitimately (i.e. not for purpose of concealing virus) encrypted files/programs usually involve encryption of the file/program as a whole, decryption of only a portion of a file/program is highly suspect and a sign that the file/program is likely infected with encrypted viral code. Once the encrypted code is detected, appropriate steps can be taken to confirm whether the encrypted code is viral and, if so, to remove the viral code before the viral code causes damage. (Page 6, line 29–page 7, line 4.)

The present disclosure provides an apparatus and methods for detecting decryption of encrypted viral code before the viral code can damage computing systems and/or infect other programs or files. An apparatus 5, according to an embodiment, is shown in FIG. 5. The apparatus includes code emulator 7 and memory monitor 9. The code emulator 7 emulates computer executable code in a subject file, preferably, on an instruction-by-instruction basis, and supplies memory access information associated with the emulated code to the memory monitor 9. The memory monitor 9 monitors the memory access information for each instruction in the emulated code and determines whether decryption is taking place. By monitoring a state of computer memory, during emulation of the subject file, the apparatus can determine whether there is sequential or quasi-sequential decryption of code in a portion of the subject file. Decryption of encrypted code in the program file often may be manifested by iterations of a code loop comprising (i) read memory, (ii) decrypt and (ii) write memory. Thus, detection of decryption may be based primarily upon analysis of changes to memory. (Page 7, lines 5-18.)

A method of detecting decryption of encrypted viral code, in accordance with an embodiment of the present disclosure, will be described with reference to FIGS. 1 and 5. Execution of code in a program or file is emulated by code emulator 7 (step S11). The memory monitor 9 monitors memory access during code emulation by the code emulator 7, and a memory area that is read during the emulation is flagged by memory monitor (step

S12). Modification, by a subsequent emulated instruction, of the flagged memory area which was read also is detected by memory monitor 9 (step S13) . (Page 7, lines 19-25.)

When each modification to memory in a group of memory accesses is to an area of memory that recently has been read, and the modifications are to sequential or quasi-sequential parts of the memory such that they form a region which is at least a predetermined size (e.g., a predetermined minimum length), then it can reliably be determined that decryption has been occurring. The memory region formed may be contiguous, evenly interspersed or otherwise quasi-contiguous. (Page 7, line 26–page 8, line 1.)

A list of memory regions modified in the emulated code execution may be maintained. More specifically, the memory monitor 9 maintains, on an instruction-by-instruction basis, a list of regions in memory where "reads" followed by "writes" have occurred during emulated execution of code in a subject file. If a memory area which was read during code emulation is modified, then the modified memory area is compared with regions in the list to determine if the modified memory area is part of one of the listed regions. If not, a new region corresponding to the modified memory area is added to the list. On the other hand, if one of the listed regions overlaps with at least a part of the modified memory area, the listed region is updated to encompass the modified memory area. When one of the regions becomes large enough, e.g., it is of such a size that it is likely that decryption is taking place or has taken place during emulation so that detection of viral code is triggered, by for example an alarm, such as a graphic alarm on a display and/or an auditory alarm through a speaker. (Page 8, lines 2-14.)

A method of detecting decryption of encrypted viral code, in accordance with this embodiment, will be described with reference to FIGS. 2 and 5. Computer executable code is emulated by the code emulator 7, preferably, one instruction at a time (step S21). The memory monitor 9 monitors the memory access information supplied by the code emulator 7 to determine whether an emulated instruction reads and/or modifies an area of memory. If memory monitor 9 determines that the emulated instruction neither reads from nor modifies an area of memory (NO, step S22), the process returns to step S21. If it is determined that an area of memory was read during emulation, the memory monitor 9 flags the area as having been read (step S23) and the process returns to step S21. If it is determined that the emulated instruction modified an area of memory and the area had been read previously, the memory monitor 9 scans through the list of regions it maintains and compares the modified memory area to the listed regions to determine if any of the listed regions overlaps at least a part of the modified memory area (step S24). If it is determined that the modified memory area is not part of any listed region (NO, step S25), then the memory monitor 9 adds the modified memory area as a new region to the list (step S26) and returns to step S21. However, if it is determined that the modified memory area overlaps an existing listed region (YES, step S25), the memory monitor 9 updates the dimensions of the memory region to encompass the modified memory area (step S27) and performs a check to determine if the size of the updated region is larger than a predetermined minimum size (step S28). If the size of the updated region is less than the predetermined minimum size (NO, step S28), then the process returns to step S21. If the size of the updated region is larger than the predetermined minimum size (YES, step S28), then decrypted code has been detected. The detection of decrypted code during the emulation triggers an alarm representing the detection of viral code. (Page 8, line 15–page 9, line 9.)

According to another embodiment, if a memory area which was read during code emulation is modified, the modified memory area is compared to the list of memory regions, one memory region at a time. If the modified memory area is not contiguous with one of the memory regions, a new memory region is added to the list of memory regions. If the modified memory area is contiguous with one of the memory regions, then that memory region is updated to encompass the modified memory area. If the size of the updated memory region is larger than a predetermined minimum size, then decrypted code has been detected. The detection of decrypted code during the emulation triggers an alarm representing the detection of viral code. (Page 9, lines 10-18.)

A method of detecting decryption of encrypted viral code, in accordance with another embodiment, will be described with reference to FIGS. 3 and 5. Code execution is emulated by the code emulator, preferably, one instruction at a time (step S31). The memory monitor 9 monitors memory access information supplied by the code emulator 7 to determine whether an emulated instruction modifies an area of memory. If it is determined that the emulated instruction did not modify the memory state (NO, step S32), then the memory monitor 9 determines whether data was read from an area of memory for the emulated instruction (step S33). If data was not read from an area of memory for the emulated instruction (NO, step S33), then the process returns to step S31. If data was read from an area of memory for the emulated instruction (YES, step S33), then the memory monitor 9 flags the memory area as having been read (step S34) and the process returns to step S31. (Page 9, lines 19-30.)

If the memory monitor 9 determines that the emulated instruction modified the memory state (YES, step S32), then it is determined whether the modified memory area had been read previously (step S35). If it is determined that the modified memory area had not been read previously (NO, step S35), then the process returns to step S31. If it is determined that the modified memory area had been read previously (YES, step S35), then one region from a list of regions is selected (step S36) and the memory monitor determines whether the modified memory area is contiguous with the selected region (step S37). If it is determined that the modified memory area is not contiguous with the selected region (NO, step S37), the memory monitor 9 then determines whether another region is on the list (step S39). If it is determined that another region exists on the list of regions and has been selected (YES, step S39), then the next region on the list of regions is selected (step S38), and then the process returns to step S37 in which it is determined whether the area is contiguous with the next region. If it is determined that there are no remaining regions on the list (NO, step S39), then the area is added as a new region to the list (step S40) and the process returns to step S31. (Page 10, lines 1-15.)

If it is determined that the area is contiguous with the selected region (YES, step S37), then the region is updated to include the area (step S41). It is then determined whether the updated region is of a sufficient size to trigger detection (step S42). If the updated region is not large enough (NO, step S42), then the process returns to step S31. If the updated region is large enough (YES, step S42), then detection of decryption is triggered. In this way, detection of decrypted viral code is effected. (Page 10, lines 16-21.)

The apparatus and methods described above may be embodied in a computer program (or some unit of code) stored on/in computer readable medium, such as memory, hard drive or removable storage media. The apparatus and methods also may be in the form of a computer data signal, in one or more segments, embodied in a transmission medium, such as the Internet, an intranet, or another (wired or wireless) transmission medium. The present disclosure also encompasses, of course, execution of the computer program stored on/in a program storage device in a computing device/system, such as, for example, shown in FIG. 4. (Page 10, lines 22-29.)

The viral code detection methodologies described herein may be performed by a computer in one or a combination of the following circumstances. The viral code detection methodologies may be performed periodically (e.g., once/twice per day) or at selected times (e.g., each time the computer is powered up or restarted) by the computer on all relevant electronic files. In addition, the viral code detection methodologies may be performed on a file (or a data stream received by the computer through a network) before the file is stored/copied/executed/opened on the computer. (Page 10, line 30–page 11, line 6.)

With regard to the independent claims currently under Appeal, Appellant provides the following concise explanation of the subject matter recited in the claim elements. For brevity, ***Appellant does not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements***. Additionally, this explanation should not be used to limit Appellant's claims but is intended to assist the Board in considering the Appeal of this Application.

For example, independent Claim 1 recites the following:

A method for detecting decryption of encrypted viral code in a subject file (*e.g.*, Figures 1-3; page 3, line 20–page 4, line 12; page 7, line 19–page 10, line 21), comprising:

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

flagging a memory area that is read during emulation of a first instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-24; page 8, lines 18-24; page 9, lines 28-30);

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 24-25; page 10, lines 1-13);

updating the list of memory regions to include the modified flagged memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 2 recites the following:

A method of detecting encrypted viral code in a subject file (*e.g.*, Figures 1-3; page 3, line 20–page 4, line 12; page 7, line 19–page 10, line 21), comprising:

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-25; page 8, lines 18-24; page 9, lines 28-30; page 10, lines 1-13);

updating the list of memory regions to include the modified memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 7 recites the following:

A program storage device readable by a machine (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), tangibly embodying a program of instructions executable by the machine to perform method steps for detecting decryption of encrypted viral code in a subject file (*e.g.*, Figures 1-3; page 3, line 20–page 4, line 12; page 7, line 19–page 10, line 21), the method steps comprising:

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

flagging a memory area that is read during emulation of a first instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-24; page 8, lines 18-24; page 9, lines 28-30);

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 24-25; page 10, lines 1-13);

updating the list of memory regions to include the modified flagged memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 8 recites the following:

A program storage device readable by a machine (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), tangibly embodying a program of instructions executable by the machine to perform method steps for detecting encrypted viral code in a subject file (*e.g.*, Figures 1-3; page 3, line 20–page 4, line 12; page 7, line 19–page 10, line 21), the method steps comprising:

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-25; page 8, lines 18-24; page 9, lines 28-30; page 10, lines 1-13);

updating the list of memory regions to include the modified memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-

11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 9 recites the following:

A computer system (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a processor (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29); and

a program storage device readable by the computer system (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), tangibly embodying a program of instructions executable by the processor to perform method steps for detecting decryption of encrypted viral code in a subject file (*e.g.*, Figures 1-3; page 3, line 20–page 4, line 12; page 7, line 19–page 10, line 21), the method steps including

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

flagging a memory area that is read during emulation of a first instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-24; page 8, lines 18-24; page 9, lines 28-30);

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 24-25; page 10, lines 1-13);

updating the list of memory regions to include the modified flagged memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 10 recites the following:

A computer system (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a processor (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29); and

a program storage device readable by the computer system (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), tangibly embodying a program of instructions executable by the computer system to perform method steps for detecting encrypted viral code (*e.g.*, Figures 1-3; page 3, lines 20–page 4, line 12; page 7, line 19–page 10, line 21), the method steps including

emulating computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21);

maintaining a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27);

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-25; page 8, lines 18-24; page 9, lines 28-30; page 10, lines 1-13);

updating the list of memory regions to include the modified memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17);

determining that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21); and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 11 recites the following:

An apparatus for detecting decryption of encrypted viral code in a subject file (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a code emulator (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26), wherein the code emulator emulates computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21), and outputs memory access information corresponding to the emulated computer executable code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26); and

a memory monitor (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), wherein the memory monitor monitors the memory access information output by the code emulator (*e.g.*, Figures 4-5; page 4, lines 17-

21; page 4, line 26–page 5, line 10), maintains a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27), flags a memory area that is read during the emulation of a first instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-24; page 8, lines 18-24; page 9, lines 28-30), detects a modification to the flagged memory area during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 24-25; page 10, lines 1-13), updates the list of memory regions to include the modified flagged memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17), determines that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21), and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 12 recites the following:

An apparatus for detecting encrypted viral code in a subject file (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a code emulator (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26), wherein the code emulator emulates computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21), and outputs memory access information corresponding to the emulated computer executable code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26); and

a memory monitor (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), wherein the memory monitor monitors the memory access information output by the code emulator (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), maintains a list of memory regions that have been read and modified during emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27), determines whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-25; page 8, lines 18-24; page 9, lines 28-30; page 10, lines 1-13), updates the list of memory regions to include the modified memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17), determines that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21), and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 17 recites the following:

A computer-readable medium which embodies instructions executable by a computer for detecting decryption of encrypted viral code in a subject file (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a first segment, including emulator code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26), wherein the emulator code emulates computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21), and outputs memory access information corresponding to the emulated computer executable code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26); and

a second segment including memory monitor code (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), wherein the memory monitor code monitors the memory access information output by the code emulator (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), maintains a list of memory regions that have been read and then modified during the emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, lines 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27), flags a memory area that is read during the emulation of a first instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-24; page 8, lines 18-24; page 9, lines 28-30), detects a modification to the flagged memory area during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 24-25; page 10, lines 1-13), updates the list of memory regions to include the modified flagged memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17), determines that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21), and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

As another example, independent Claim 18 recites the following:

A computer-readable medium which embodies instructions executable by a computer for detecting encrypted viral code in a subject file (*e.g.*, Figures 4-5; page 6, lines 5-15; page 10, lines 22-29), comprising:

a first segment including emulator code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26), wherein the emulator code emulates computer executable code in a subject file (*e.g.*, Figures 1-3; page 3, lines 21-22; page 3, line 28–page 4, line 1; page 7, line 21; page 8, lines 16-17; page 9, lines 20-21), and outputs memory access information corresponding to the emulated computer executable code (*e.g.*, Figures 4-5; page 4, lines 13-17, 22-26); and

a second segment including memory monitor code (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), wherein the memory monitor code monitors the memory access information output by the code emulator (*e.g.*, Figures 4-5; page 4, lines 17-21; page 4, line 26–page 5, line 10), maintains a list of memory regions that have been read and modified during emulation (*e.g.*, Figures 1-3; page 3, lines 22-23; page 7, line 26–page 8, line 5; page 8, lines 24-28; page 9, lines 21-27), determines whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code (*e.g.*, Figures 1-3; page 3, lines 23-26; page 7, lines 21-25; page 8, lines 18-24; page 9, lines 28-30; page 10, lines 1-13), updates the list of memory regions to include the modified memory area (*e.g.*, Figures 1-3; page 3, lines 26-27; page 4, lines 2-12; page 8, lines 5-11; page 8, line 28–page 9, line 4; page 9, lines 10-15; page 10, lines 13-17), determines that one of the listed memory regions is larger than a predetermined size (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 8, line 30–page 9, line 7, page 9, lines 15-17; page 10, lines 17-21), and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code (*e.g.*, Figures 1-3; page 3, lines 27-28; page 8, lines 11-14; page 9, lines 7-9, 17-18; page 10, lines 20-21).

Grounds of Rejection to be Reviewed on Appeal

1. Are Claims 1, 2, 7-12, and 17-18 patentable under 35 U.S.C. § 101?

2. Are Claims 1, 2, 7-12, and 17-18 patentable under 35 U.S.C. § 112?

3. a. Are Claims 1-4, 6-14, and 16-18 patentable over the Examiner's proposed combination of U.S. Patent No. 6,357,008 to Nachenberg ("*Nachenberg I*") and U.S. Patent No. 6,453,345 to Trcka ("*Trcka*") under 35 U.S.C. § 103(a)?

b. Are Claims 5 and 15 patentable over the Examiner's proposed combination of *Nachenberg I* and *Trcka* and U.S. Patent No. 6,971,019 to Nachenberg ("*Nachenberg II*") under 35 U.S.C. § 103(a)?

Grouping of Claims

Appellant has made an effort to group claims to reduce the burden on the Board. In the Argument section of this Appeal Brief, where appropriate, Appellant presents arguments as to why particular claims subject to a ground of rejection are separately patentable from other claims subject to the same ground of rejection. To reduce the number of groups and thereby reduce the burden on the Board, Appellant does not individually argue every claim that recites patentable distinctions over the references cited by the Examiner, particularly in light of the clear allowability of Appellant's independent claims.

The claims of each group provided below may be deemed to stand or fall together for purposes of this Appeal.

With regard to the ground of rejection identified as issue 1 above, the claims subject to that ground of rejection may be grouped together as follows for purposes of this Appeal:

1. Group 1 may include Claims 1-18.

<u>Argument</u>

I. The Claims are Patentable under 35 U.S.C. § 101

Claims 17-18 have been amended to correct an informality pointed out by the Examiner. Appellant thanks the Examiner for pointing out the informality. Accordingly, Claims 17-18 are allowable under 35 U.S.C. § 101.

II. The Claims are Patentable under 35 U.S.C. § 112

Claims 1, 2, 7-12, and 17-18 have been amended to correct an informality pointed out by the Examiner. Appellant thanks the Examiner for pointing out the informality. Accordingly, Claims 1, 2, 7-12, and 17-18 are allowable under 35 U.S.C. § 112.

III. The Claims are Patentable under 35 U.S.C. § 103(a)

A. Claims 1-4, 6-14, and 16-18 are Patentable Over the *Nachenberg I-Trcka* Combination Proposed by the Examiner

Claims 1-4, 6-14, and 16-18 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the *Nachenberg I-Trcka* combination. Copies of *Nachenberg I* and *Trcka* are included in the Evidence Appendix. Appellant respectfully submits that the Examiner's proposed *Nachenberg I-Trcka* combination fails to support the obviousness rejections of these claims. Appellant respectfully submits that these rejections are therefore improper and should be reversed by the Board.

1. Standard

The question raised under 35 U.S.C. § 103 is whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art at the time of the invention. *See* 35 U.S.C. § 103(a). Accordingly, even if all elements of a claim are disclosed in various prior art references, which is certainly not the case here as discussed below, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill in the art at the time of the invention would have been prompted to modify the teachings of a reference or combine the teachings of multiple references to arrive at the claimed invention.

The M.P.E.P. sets forth the strict legal standard for establishing a *prima facie* case of obviousness based on modification or combination of prior art references. "To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references where combined) must teach or suggest all the claim limitations." M.P.E.P. § 2142, 2143. The teaching, suggestion or motivation for the modification or combination and the reasonable expectation of success must both be found in the prior art and cannot be based on an applicant's disclosure. *See Id.* (citations omitted). "Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art" at the time of the invention. M.P.E.P. § 2143.01. Even the fact that references *can* be modified or combined does not render the resultant modification or combination obvious unless the prior art teaches or suggests the desirability of the modification or combination. *See Id.* (citations omitted). Moreover, "To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. All words in a claim must be considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03 (citations omitted).

The governing Federal Circuit case law makes this strict legal standard even more clear.[1] According to the Federal Circuit, "a showing of a suggestion, teaching, or motivation to combine or modify prior art references is an essential component of an obviousness holding." *In re Sang-Su Lee*, 277 F.3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1433 (Fed. Cir. 2002) (quoting *Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1124-25, 56 U.S.P.Q.2d 1456, 1459 (Fed. Cir. 2000)). "Evidence of a suggestion, teaching,

[1] Note M.P.E.P. 2145 X.C. ("The Federal Circuit has produced a number of decisions overturning obviousness rejections due to a lack of suggestion in the prior art of the desirability of combining references.").

or motivation . . . may flow from the prior art references themselves, the knowledge of one of ordinary skill in the art, or, in some cases, the nature of the problem to be solved." *In re Dembiczak*, 175 F.3d 994, 999, 50 U.S.P.Q.2d 1614, 1617 (Fed. Cir. 1999). However, the "range of sources available ... does not diminish the requirement for actual evidence." *Id.* Although a prior art device "may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so." *In re Mills*, 916 F.2d at 682, 16 U.S.P.Q.2d at 1432. *See also In re Rouffet,* 149 F.3d 1350, 1357, 47 U.S.P.Q.2d 1453, 1457-58 (Fed. Cir. 1998) (holding a *prima facie* case of obviousness not made where the combination of the references taught every element of the claimed invention but did not provide a motivation to combine); *In Re Jones*, 958 F.2d 347, 351, 21 U.S.P.Q.2d 1941, 1944 (Fed. Cir. 1992) ("Conspicuously missing from this record is any evidence, other than the PTO's speculation (if that can be called evidence) that one of ordinary skill in the herbicidal art would have been motivated to make the modification of the prior art salts necessary to arrive at" the claimed invention.). Even a determination that it would have been obvious to one of ordinary skill in the art at the time of the invention to try the proposed modification or combination is not sufficient to establish a *prima facie* case of obviousness. *See In re Fine*, 837 F.2d 1071, 1075, 5 U.S.P.Q.2d 1596, 1599 (Fed. Cir. 1988).

In addition, the M.P.E.P. and the Federal Circuit repeatedly warn against using an applicant's disclosure as a blueprint to reconstruct the claimed invention. For example, the M.P.E.P. states, "The tendency to resort to 'hindsight' based upon applicant's disclosure is often difficult to avoid due to the very nature of the examination process. However, impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art." M.P.E.P. § 2142. The governing Federal Circuit cases are equally clear. "A critical step in analyzing the patentability of claims pursuant to [35 U.S.C. § 103] is casting the mind back to the time of invention, to consider the thinking of one of ordinary skill in the art, guided only by the prior art references and the then-accepted wisdom in the field. ... Close adherence to this methodology is especially important in cases where the very ease with which the invention can be understood may prompt one 'to fall victim to the insidious effect of a hindsight syndrome wherein that which only the invention taught is used against its teacher.'" *In re Kotzab,* 217 F.3d 1365, 1369, 55 U.S.P.Q.2d 1313, 1316 (Fed. Cir. 2000) (citations omitted). In *In re Kotzab*, the Federal

Circuit noted that to prevent the use of hindsight based on the invention to defeat patentability of the invention, the court requires the Examiner to show a motivation to combine the references that create the case of obviousness. *See id. See also, e.g., Grain Processing Corp. v. American Maize-Products,* 840 F.2d 902, 907, 5 U.S.P.Q.2d 1788, 1792 (Fed. Cir. 1988). Similarly, in *In re Dembiczak*, the Federal Circuit reversed a finding of obviousness by the Board, explaining that the required evidence of such a teaching, suggestion, or motivation is essential to avoid impermissible hindsight reconstruction of an applicant's invention:

> Our case law makes clear that the best defense against the subtle but powerful attraction of hind-sight obviousness analysis is *rigorous application of the requirement for a showing of the teaching or motivation to combine prior art references.* Combining prior art references without evidence of such a suggestion, teaching, or motivation simply takes the inventor's disclosure as a blueprint for piecing together the prior art to defeat patentability—the essence of hindsight.

175 F.3d at 999, 50 U.S.P.Q.2d at 1617 (emphasis added) (citations omitted).

2. *The Nachenberg I Reference*

Nachenberg I discloses detecting computer viruses ***based on suspicious behavior***. According to *Nachenberg I*:

> [The method comprises] three phases: a decryption phase, an exploration phase, and an evaluation phase. A purpose of the decryption phase is to emulate a sufficient number of instructions to allow an encrypted virus to decrypt its viral body. A purpose of the exploration phase is to emulate at least once all sections of code within a region deemed likely to contain any virus present in the target program. ***A purpose of the evaluation phase is to analyze any suspicious behavior observed during the decryption and exploration phases to determine whether the target appears to be infected.***

(*Nachenberg I*, Abstract, emphasis added.) That is, the *Nachenberg I* method ***analyzes suspicious behavior*** to detect computer viruses.

Nachenberg I discloses establishing whether a region of a certain size has been decrypted in order to determine when to move from the decryption phase. According to *Nachenberg I*:

> On the other hand, if the first threshold number has been reached, then the decryption module 152 determines in a fourth procedure 308 whether a region of a certain minimum size or larger appears to have been decrypted. ... ***If no such region appears to have been decrypted***, then under the assumption that any virus present is unlikely to be an encrypted virus, ***the decryption phase 252 ends*** and the exploration phase 254 begins.
>
> On the other hand, ***if such a region appears to have been decrypted***, then emulation in ***the decryption phase 252 continues*** to allow further decryption by fetching the instruction at the virtual CS:IP in the sixth procedure 312 unless a second threshold number of emulated instructions has been reached.

(*Nachenberg I*, column 8, lines 1-19, emphasis added.) That is, *Nachenberg I* establishes whether a region of a certain size has been decrypted merely to determine when to move from the decryption phase, ***but not to detect computer viruses***. In summary, *Nachenberg I* detects computer viruses ***based on suspicious behavior, but not based on establishing whether a region of a certain size has been decrypted***.

3. The Trcka Reference

Trcka discloses an automated monitor application. According to *Trcka*:

> The Automated Monitor application 140 uses known data processing techniques (virus checking, transaction monitoring, etc.) to automatically check for and track suspect network events. In one configuration option, the Automated Monitor 140 checks all inbound transfers of executable files for known viruses. By selecting an ALERT MONITOR menu option on the graphical user interface 104, the user can enable and disable various visual and audible event alarms. For example, the user can configure the Automated Monitor 140 to trigger an audible or visual alarm upon detecting a virus in an inbound file transfer.

(*Trcka*, column 17, lines 24-34.) That is, *Trcka* merely discloses ***triggering an alarm upon detecting a known virus***.

4. The Proposed Nachenberg I-Trcka Combination Fails to Disclose, Teach, or Suggest the Elements Recited in Claims 1-4, 6-14, and 16-18

Appellant respectfully submits that the *Nachenberg I-Trcka* combination proposed by the Examiner fails to disclose, teach, or suggest "triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size," as recited in Claim 2.

First, *Nachenberg I* fails to disclose, teach, or suggest the element. As discussed above, *Nachenberg I* detects computer viruses ***based on suspicious behavior, but not based on establishing whether a region of a certain size has been decrypted***. Presumably, if the *Nachenberg I* method triggered an alarm, the alarm would be triggered in response to ***detecting suspicious behavior***, but not in response to establishing whether a region of a certain size has been decrypted. Accordingly, *Nachenberg I* fails to disclose, teach, or suggest "triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size," as recited Claim 2.

Second, *Trcka* fails to disclose, teach, or suggest the element. *Trcka* merely discloses ***triggering an alarm upon detecting a known virus***. *Trcka*, however, fails to disclose, teach, or suggest "triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size," as recited Claim 2.

Consequently, *Nachenberg I-Trcka* combination proposed by the Examiner fails to disclose, teach, or suggest "triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size," recited Claim 2. Accordingly, Claim 2 is allowable over the *Nachenberg I-Trcka* combination. For analogous reasons, Claims 1, 7-12, and 17-18 are allowable over the *Nachenberg I-Trcka* combination.

For at least these reasons, Appellant respectfully submits that the rejection of independent Claims 1, 2, 7-12, and 17-18, and their dependent claims is improper and should be reversed by the Board.

B. Claims 5 and 15 are Patentable over the Proposed *Nachenberg I-Trcka-Nachenberg II* Combination Proposed by the Examiner

For the reasons discussed above, Claims 5 and 15 are allowable over the *Nachenberg I-Trcka* combination, even when combined with *Nachenberg II* as proposed by the Examiner. For at least these reasons, Appellant respectfully submits that the rejection of independent Claims 5 and 15 is improper and should be reversed by the Board.

Claims Appendix

IN THE CLAIMS:

1. (Previously Presented) A method for detecting decryption of encrypted viral code in a subject file, comprising:

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

flagging a memory area that is read during emulation of a first instruction in the computer executable code;

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified flagged memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

2. (Previously Presented) A method of detecting encrypted viral code in a subject file, comprising:

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

3. (Original) The method of claim 2, wherein the emulation is performed on an instruction-by-instruction basis.

4. (Original) The method of claim 2, further comprising:

determining whether a selected one of the listed memory regions overlaps the modified memory area; and

updating the selected memory region to encompass the modified memory area.

5. (Original) The method of claim 2, further comprising:

determining whether a selected one of the listed memory regions is contiguous with the modified memory area; and

updating the selected memory region to encompass the modified memory area.

6. (Original) The method of claim 2, further comprising:

determining whether the modified memory area overlaps the listed memory regions; and

adding the modified memory area as a new memory region to the list of memory regions, if the modified memory area does not overlap any of the listed memory regions.

7. (Previously Presented) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting decryption of encrypted viral code in a subject file, the method steps comprising:

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

flagging a memory area that is read during emulation of a first instruction in the computer executable code;

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified flagged memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

8. (Previously Presented) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting encrypted viral code in a subject file, the method steps comprising:

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

9. (Previously Presented) A computer system, comprising:

a processor; and

a program storage device readable by the computer system, tangibly embodying a program of instructions executable by the processor to perform method steps for detecting decryption of encrypted viral code in a subject file, the method steps including

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

flagging a memory area that is read during emulation of a first instruction in the computer executable code;

detecting a modification to the flagged memory area during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified flagged memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

10. (Previously Presented) A computer system, comprising:

a processor; and

a program storage device readable by the computer system, tangibly embodying a program of instructions executable by the computer system to perform method steps for detecting encrypted viral code, the method steps including

emulating computer executable code in a subject file;

maintaining a list of memory regions that have been read and then modified during the emulation;

determining whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code;

updating the list of memory regions to include the modified memory area;

determining that one of the listed memory regions is larger than a predetermined size; and

triggering a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

11. (Previously Presented) An apparatus for detecting decryption of encrypted viral code in a subject file, comprising:

a code emulator, wherein the code emulator emulates computer executable code in a subject file, and outputs memory access information corresponding to the emulated computer executable code; and

a memory monitor, wherein the memory monitor monitors the memory access information output by the code emulator, maintains a list of memory regions that have been read and then modified during the emulation, flags a memory area that is read during the emulation of a first instruction in the computer executable code, detects a modification to the flagged memory area during emulation of a second instruction in the computer executable code, updates the list of memory regions to include the modified flagged memory area, determines that one of the listed memory regions is larger than a predetermined size, and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

12. (Previously Presented) An apparatus for detecting encrypted viral code in a subject file, comprising:

a code emulator, wherein the code emulator emulates computer executable code in a subject file, and outputs memory access information corresponding to the emulated computer executable code; and

a memory monitor, wherein the memory monitor monitors the memory access information output by the code emulator, maintains a list of memory regions that have been read and modified during emulation, determines whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code, updates the list of memory regions to include the modified memory area, determines that one of the listed memory regions is larger than a predetermined size, and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

13. (Original) The apparatus of claim 12, wherein the code emulator performs the emulation on an instruction-by-instruction basis.

14. (Original) The apparatus of claim 12, wherein the memory monitor determines whether a selected one of the listed memory regions overlaps the modified memory area, and updates the selected memory region to encompass the modified memory area.

15. (Original) The apparatus of claim 12, wherein the memory monitor determines whether a selected one of the listed memory regions is contiguous with the modified memory area, and updates the selected memory region to encompass the modified memory area.

16. (Original) The apparatus of claim 12, wherein the memory monitor determines whether the modified memory area does not overlap the listed memory regions, and adds the modified memory area as a new memory region to the list of memory regions.

17. (Previously Presented) A medium which embodies instructions executable by a computer for detecting decryption of encrypted viral code in a subject file, comprising:

a first segment, including emulator code, wherein the emulator code emulates computer executable code in a subject file, and outputs memory access information corresponding to the emulated computer executable code; and

a second segment including memory monitor code, wherein the memory monitor code monitors the memory access information output by the code emulator, maintains a list of memory regions that have been read and then modified during the emulation, flags a memory area that is read during the emulation of a first instruction in the computer executable code, detects a modification to the flagged memory area during emulation of a second instruction in the computer executable code, updates the list of memory regions to include the modified flagged memory area, determines that one of the listed memory regions is larger than a predetermined size, and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

18. (Previously Presented) A medium which embodies instructions executable by a computer for detecting encrypted viral code in a subject file, comprising:

a first segment including emulator code, wherein the emulator code emulates computer executable code in a subject file, and outputs memory access information corresponding to the emulated computer executable code; and

a second segment including memory monitor code, wherein the memory monitor code monitors the memory access information output by the code emulator, maintains a list of memory regions that have been read and modified during emulation, determines whether a memory area is read during emulation of a first instruction in the computer executable code and whether the memory area is modified during emulation of a second instruction in the computer executable code, updates the list of memory regions to include the modified memory area, determines that one of the listed memory regions is larger than a predetermined size, and triggers a viral detection alarm in response to determining that one of the listed memory regions is larger than the predetermined size, the viral detection alarm indicating detection of viral code.

Evidence Appendix

Other than the references attached to this Appeal Brief in this Evidence Appendix, no evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, and no other evidence was entered by the Examiner and relied upon by Appellant in the Appeal.

Related Proceedings Appendix

None.